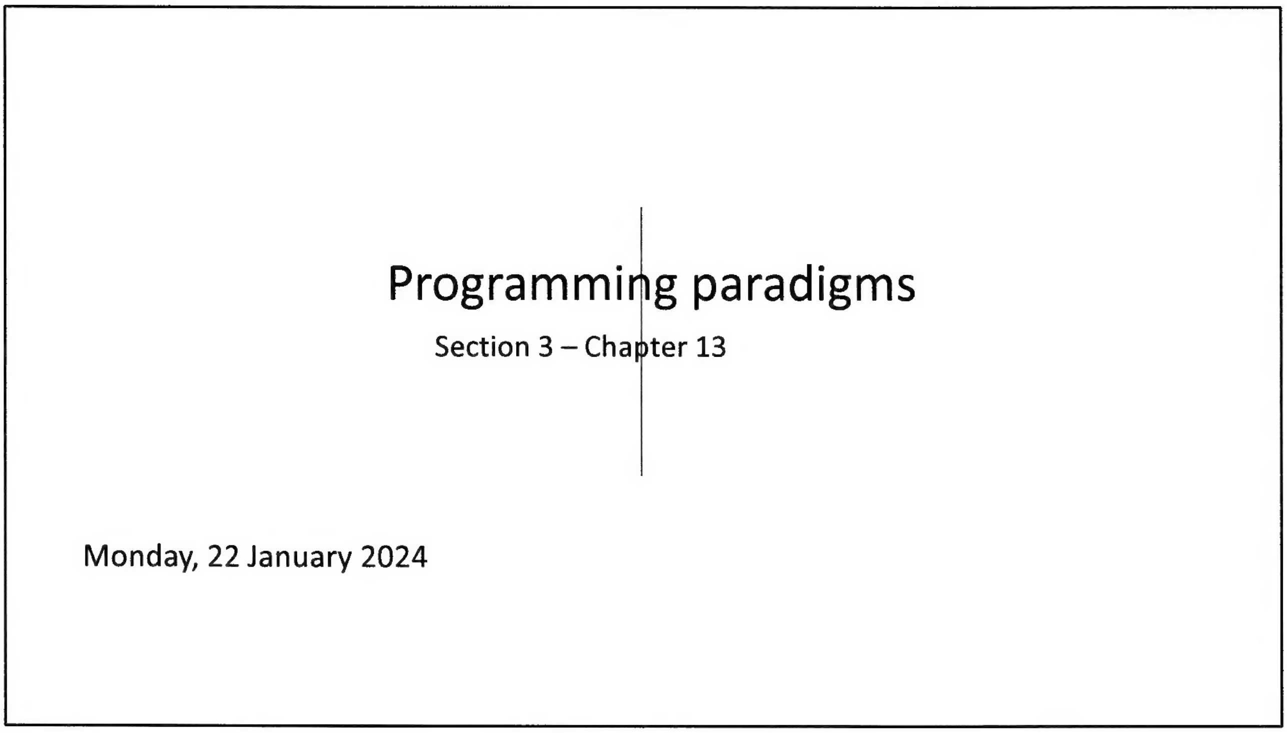
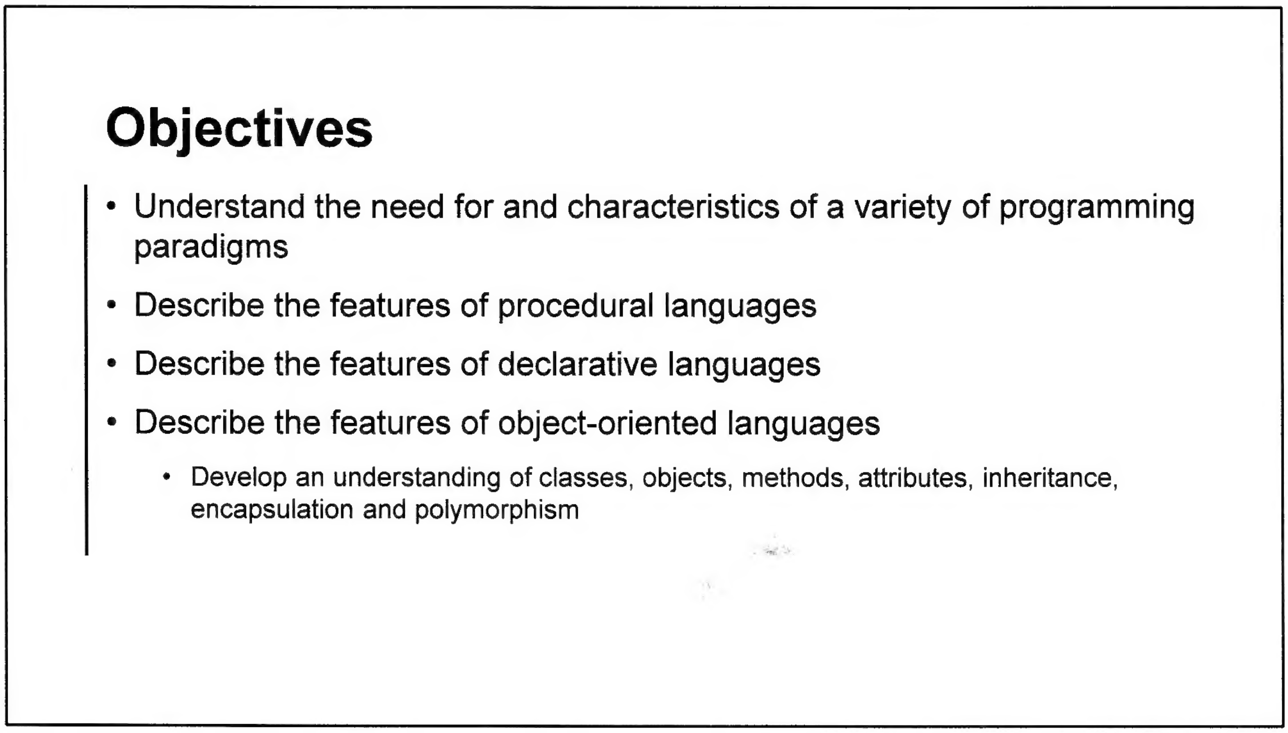


Programming paradigms

Section 3 – Chapter 13

Monday, 22 January 2024



Objectives

- Understand the need for and characteristics of a variety of programming paradigms
- Describe the features of procedural languages
- Describe the features of declarative languages
- Describe the features of object-oriented languages
  - Develop an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism

Key Words

- Programming Paradigm –
  - a style or way of programming
- Object Orientated Language –
  - Developed to make it possible to abstract details of implementation away from the user
- Encapsulation –
  - Attributes and methods are wrapped into a single entity
- Polymorphism –
  - An inherited class may have methods and attributes that do not exist in the parent class

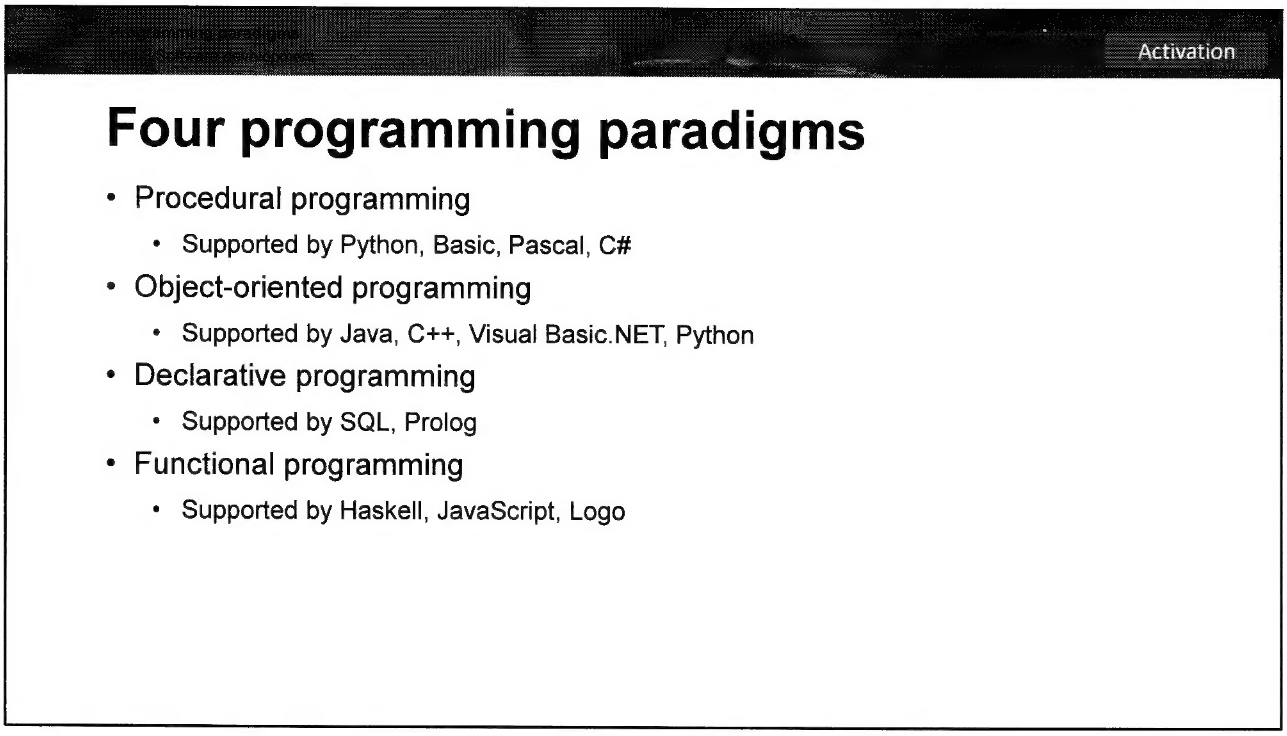
Programming paradigms

Software development

Connection

Programming languages

- Which programming language are you most familiar with?
- How would you describe the language – procedural? Object-oriented?
- How many different types of high-level programming language can you think of?

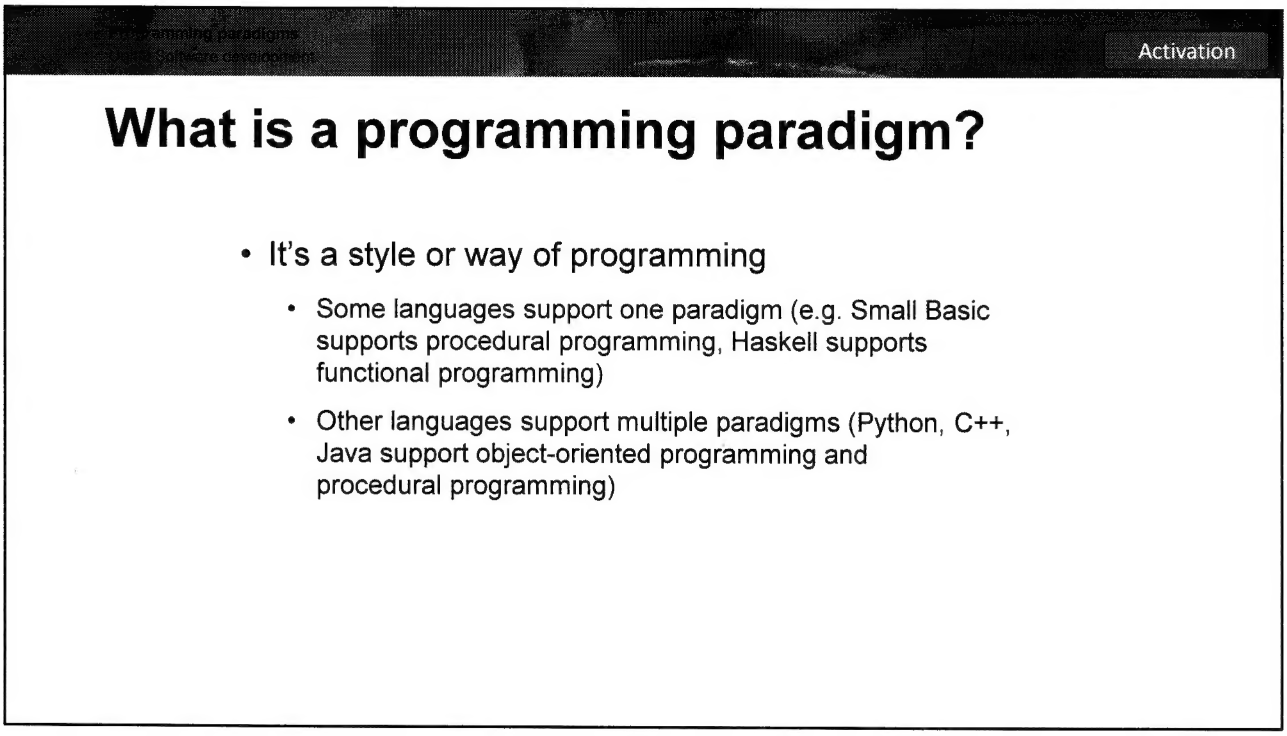


Programming paradigms
Unit [illegible] Software development

Activation

Four programming paradigms

- Procedural programming
  - Supported by Python, Basic, Pascal, C#
- Object-oriented programming
  - Supported by Java, C++, Visual Basic.NET, Python
- Declarative programming
  - Supported by SQL, Prolog
- Functional programming
  - Supported by Haskell, JavaScript, Logo

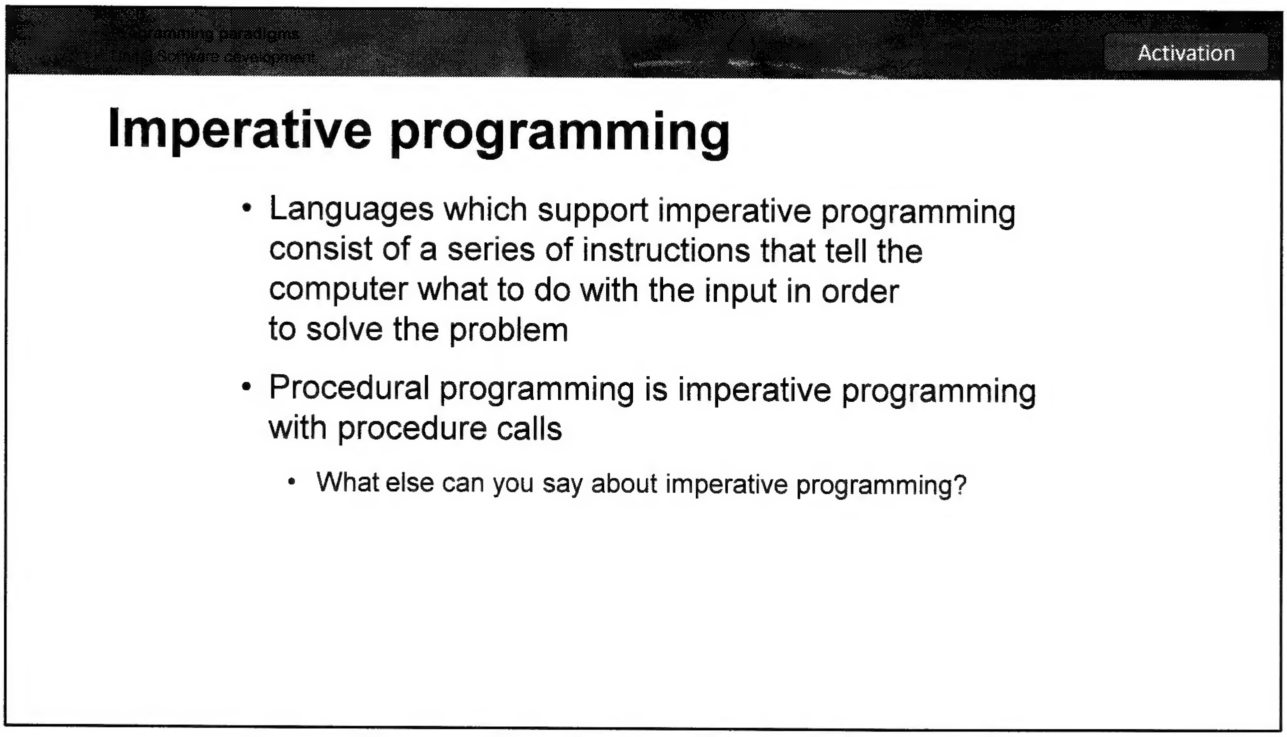


Programming paradigms
Unit [illegible] Software development

Activation

What is a programming paradigm?

- It's a style or way of programming
  - Some languages support one paradigm (e.g. Small Basic supports procedural programming, Haskell supports functional programming)
  - Other languages support multiple paradigms (Python, C++, Java support object-oriented programming and procedural programming)

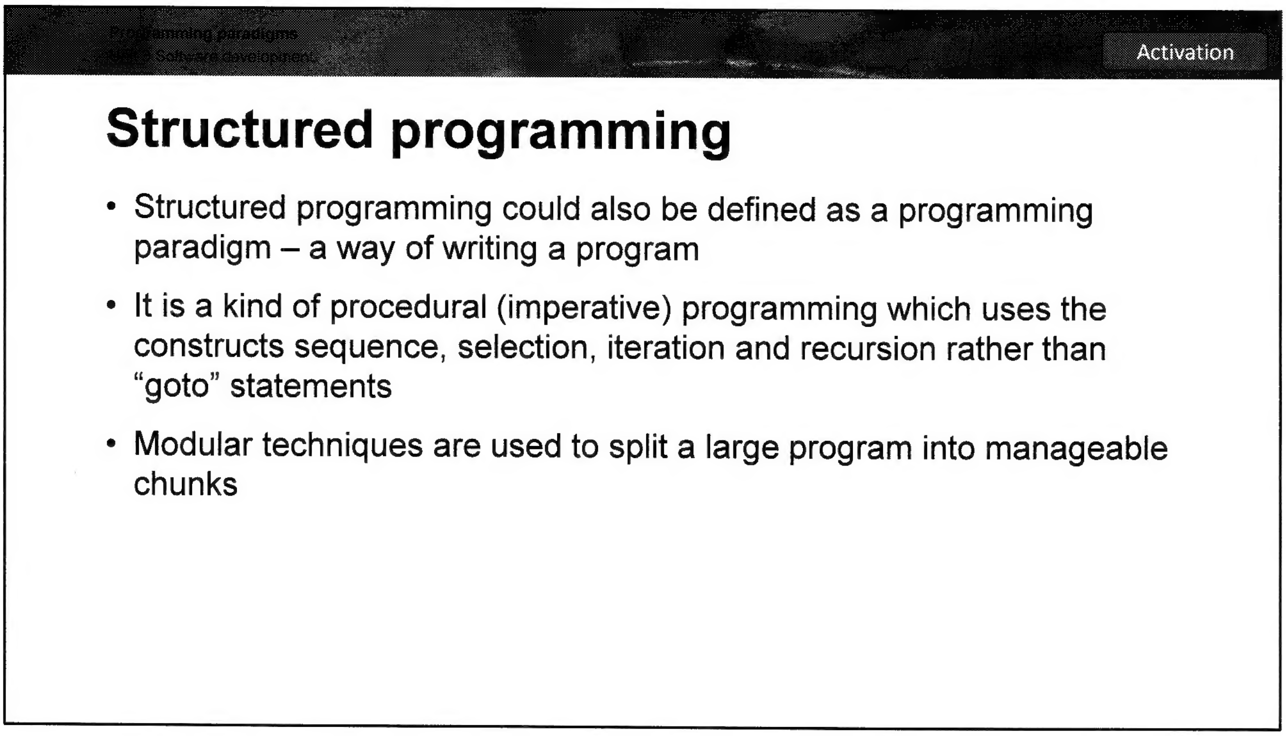


Programming paradigms
Unit [illegible] Software development

Activation

Imperative programming

- Languages which support imperative programming consist of a series of instructions that tell the computer what to do with the input in order to solve the problem
- Procedural programming is imperative programming with procedure calls
  - What else can you say about imperative programming?

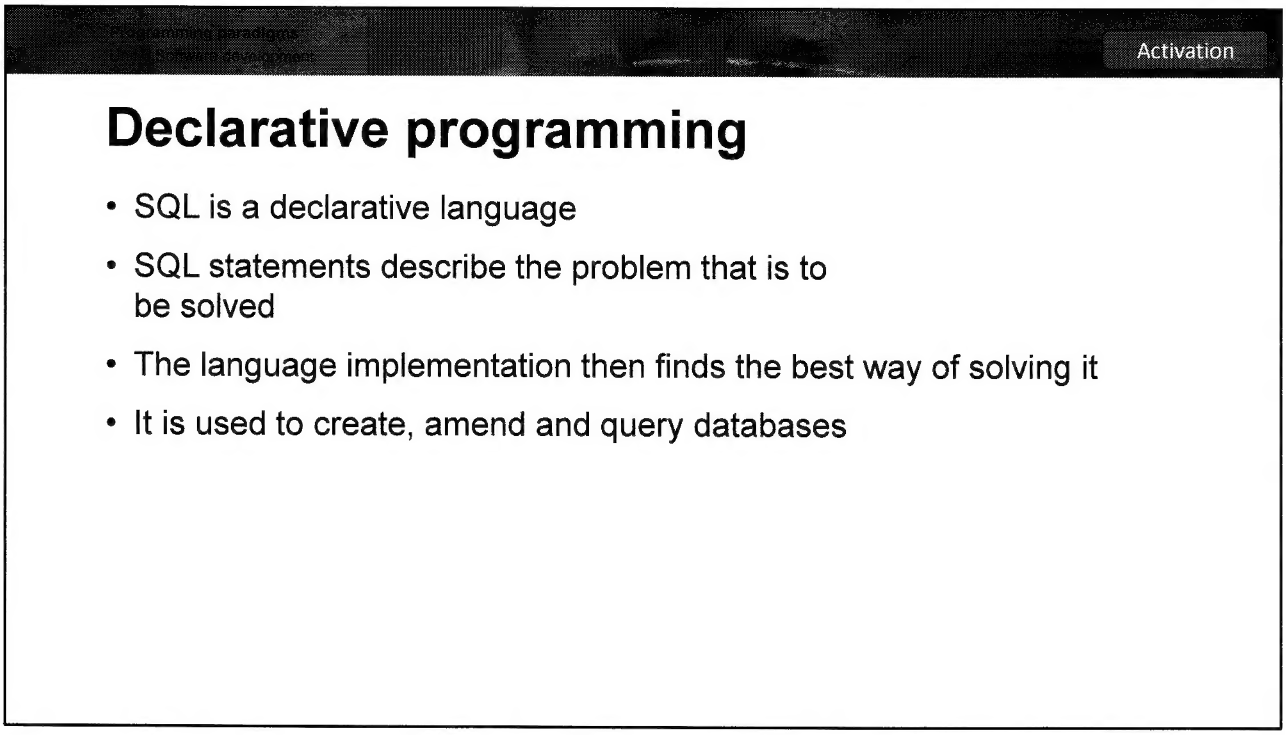


Programming paradigms
Unit [illegible] Software development

Activation

Structured programming

- Structured programming could also be defined as a programming paradigm – a way of writing a program
- It is a kind of procedural (imperative) programming which uses the constructs sequence, selection, iteration and recursion rather than "goto" statements
- Modular techniques are used to split a large program into manageable chunks

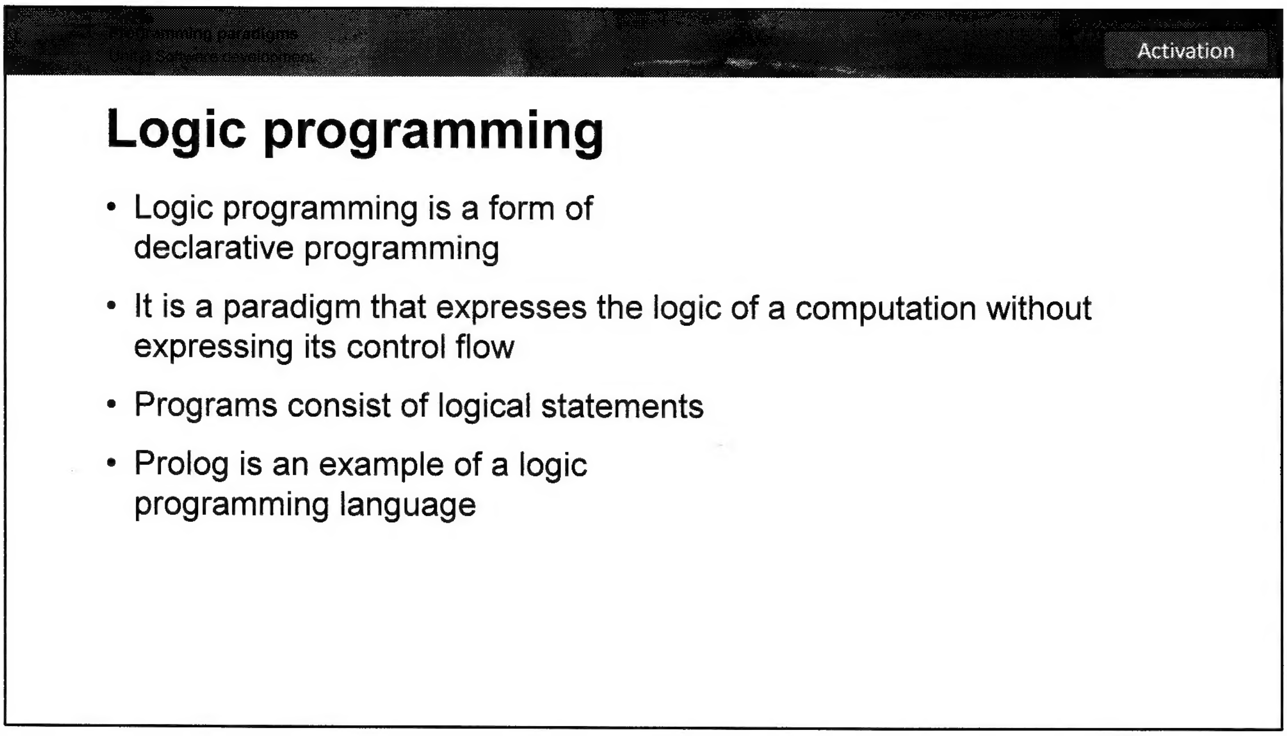


Programming paradigms
Unit 3 Software development

Activation

Declarative programming

- SQL is a declarative language
- SQL statements describe the problem that is to be solved
- The language implementation then finds the best way of solving it
- It is used to create, amend and query databases

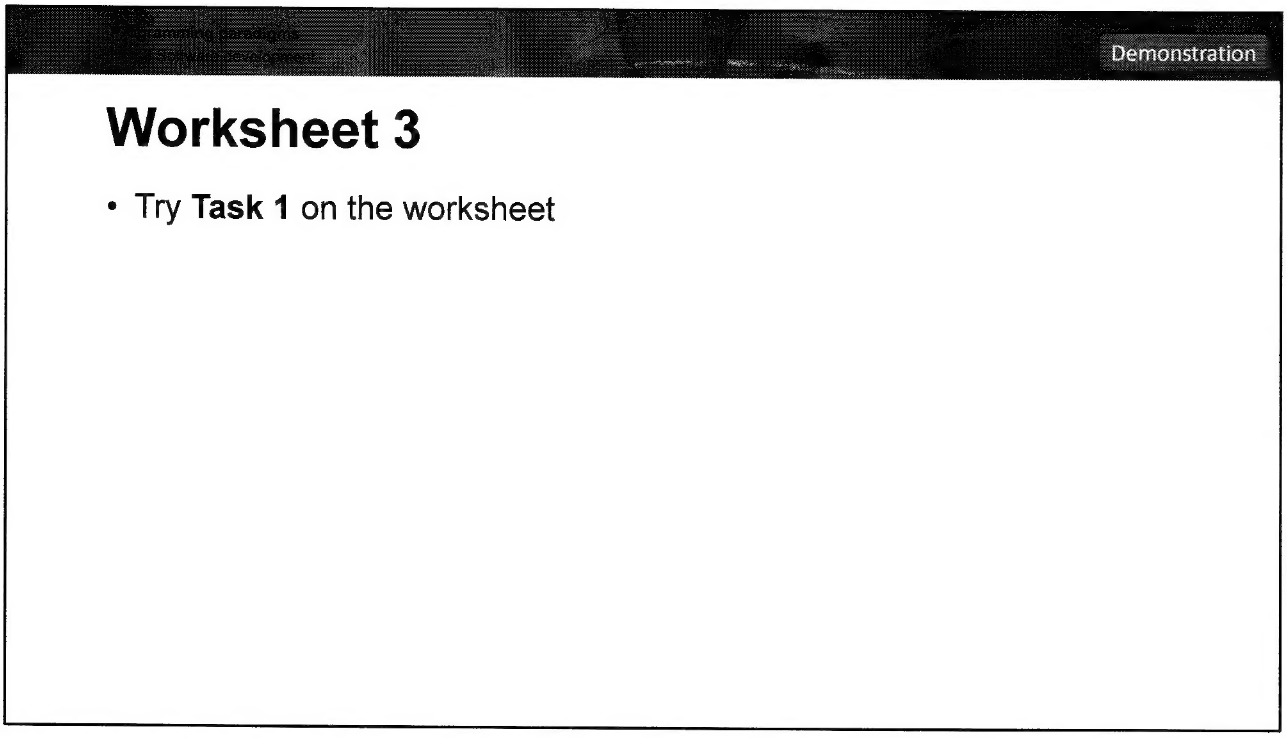


Programming paradigms
Unit 3 Software development

Activation

Logic programming

- Logic programming is a form of declarative programming
- It is a paradigm that expresses the logic of a computation without expressing its control flow
- Programs consist of logical statements
- Prolog is an example of a logic programming language

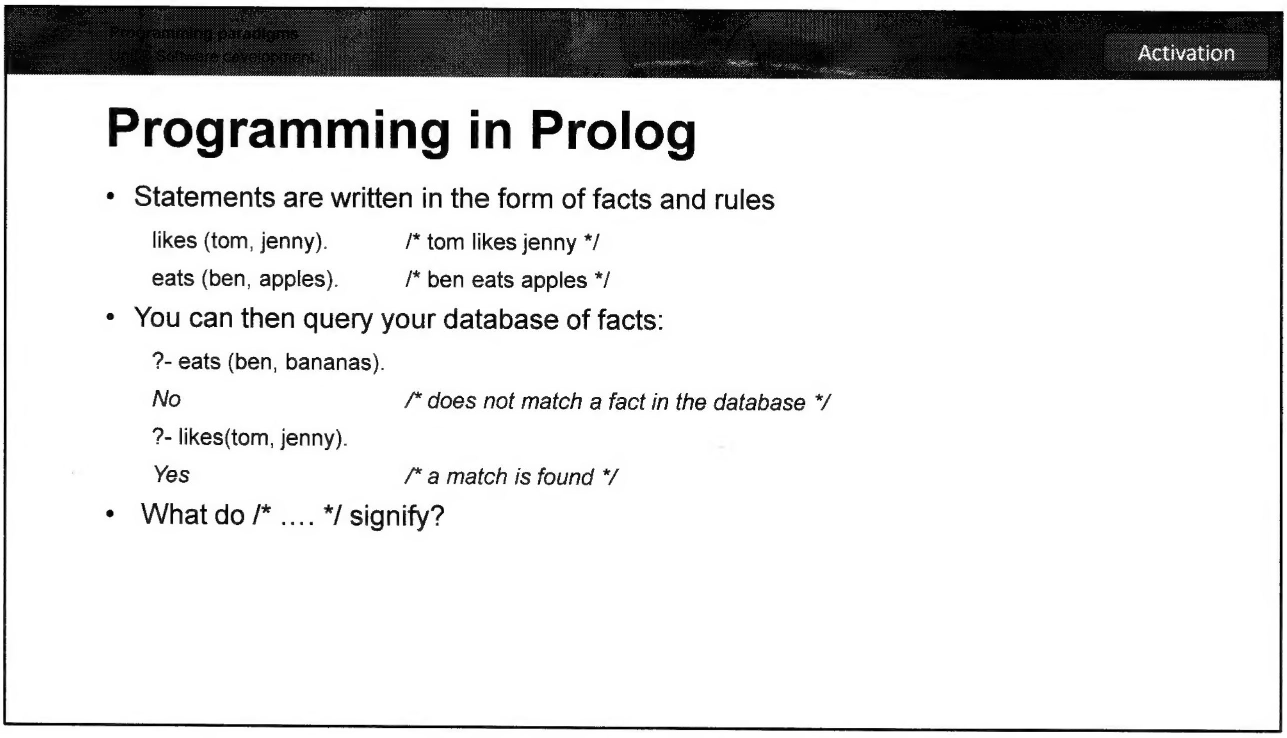


Programming paradigms
Unit [illegible] Software development

Demonstration

Worksheet 3

- Try **Task 1** on the worksheet



Programming paradigms
Unit [illegible] Software development

Activation

Programming in Prolog

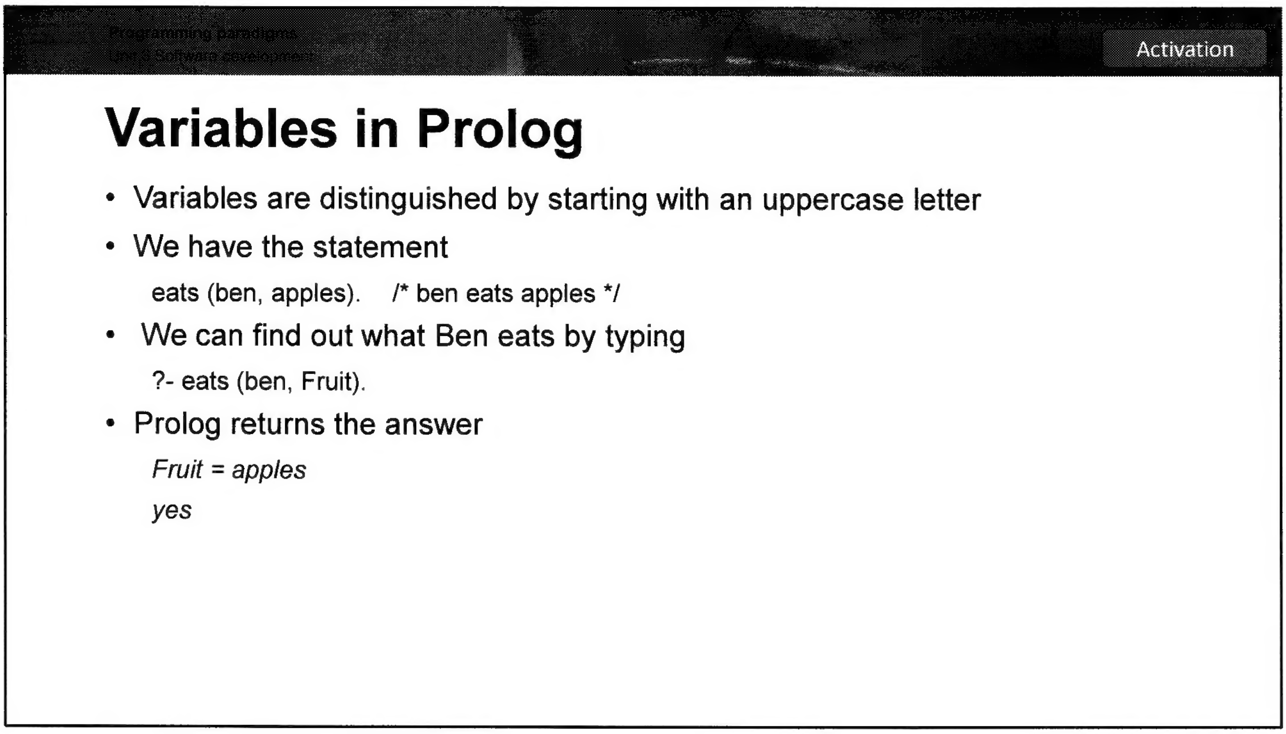
- Statements are written in the form of facts and rules

```
likes (tom, jenny).        /* tom likes jenny */
eats (ben, apples).        /* ben eats apples */
```

- You can then query your database of facts:

```
?- eats (ben, bananas).
No                         /* does not match a fact in the database */
?- likes(tom, jenny).
Yes                        /* a match is found */
```

- What do /* …. */ signify?



Variables in Prolog

- Variables are distinguished by starting with an uppercase letter
- We have the statement

```
eats (ben, apples).    /* ben eats apples */
```

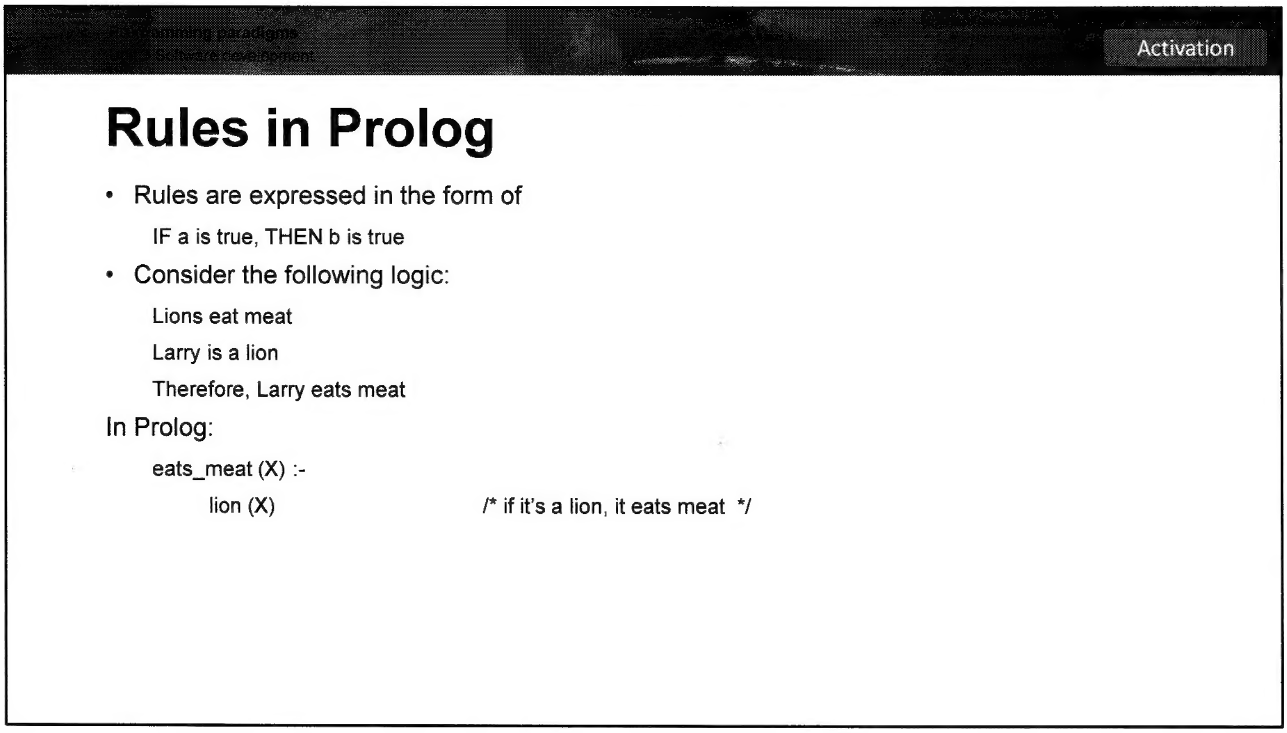
- We can find out what Ben eats by typing

```
?- eats (ben, Fruit).
```

- Prolog returns the answer

Fruit = apples

yes



Rules in Prolog

- Rules are expressed in the form of

  IF a is true, THEN b is true

- Consider the following logic:

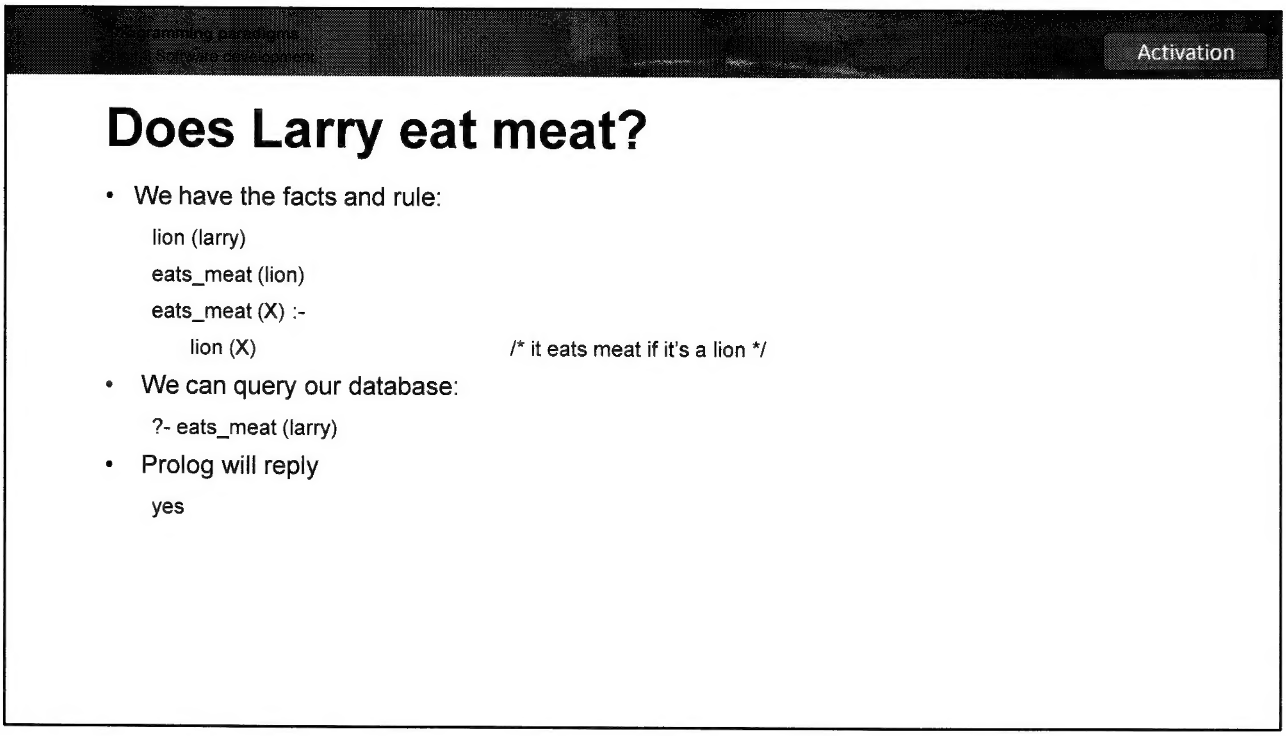
  Lions eat meat

  Larry is a lion

  Therefore, Larry eats meat

In Prolog:

```
eats_meat (X) :-
    lion (X)                    /* if it's a lion, it eats meat */
```



Activation

Does Larry eat meat?

- We have the facts and rule:

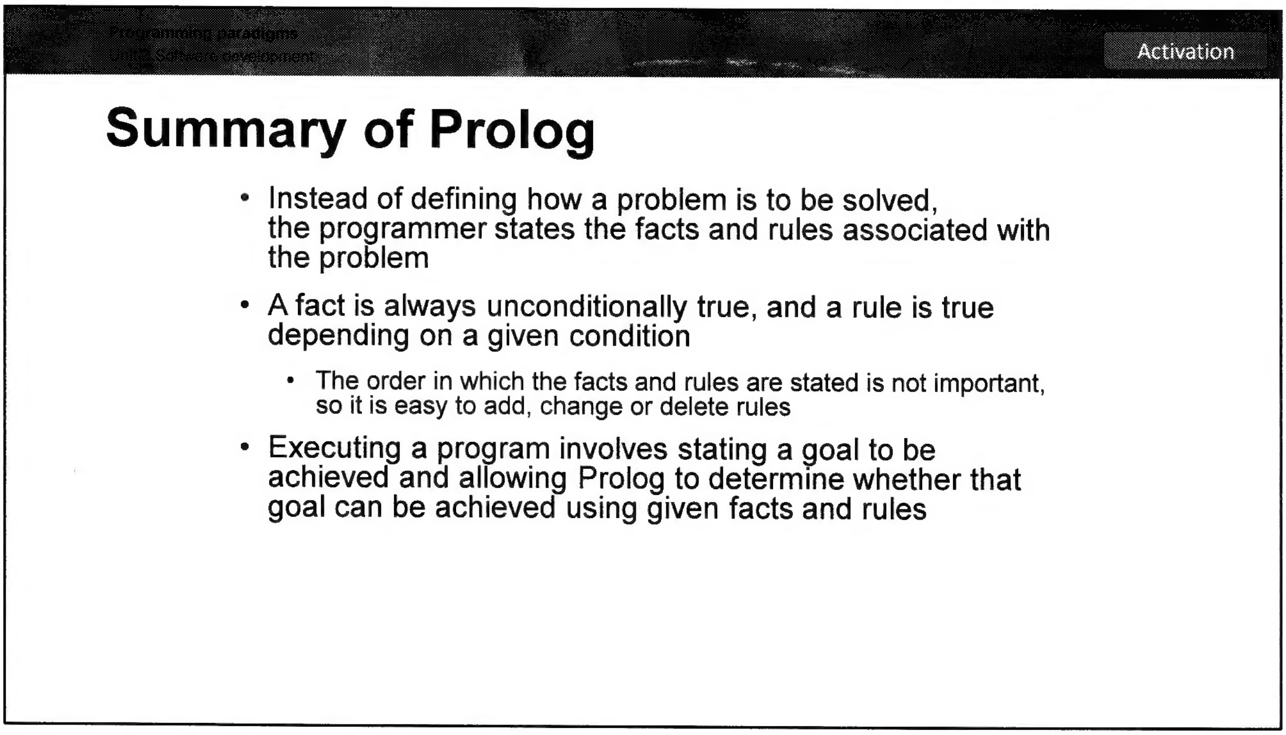
```
lion (larry)
eats_meat (lion)
eats_meat (X) :-
    lion (X)                    /* it eats meat if it's a lion */
```

- We can query our database:

```
?- eats_meat (larry)
```

- Prolog will reply

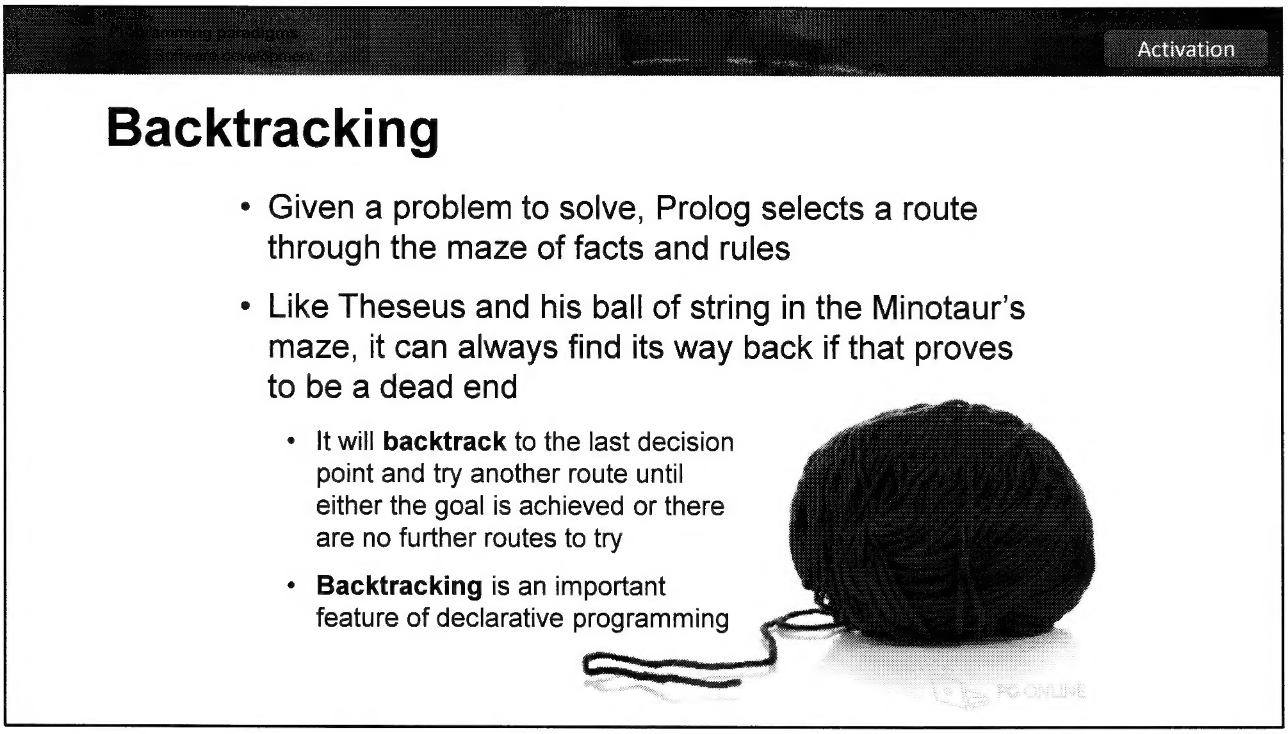
```
yes
```



Activation

Summary of Prolog

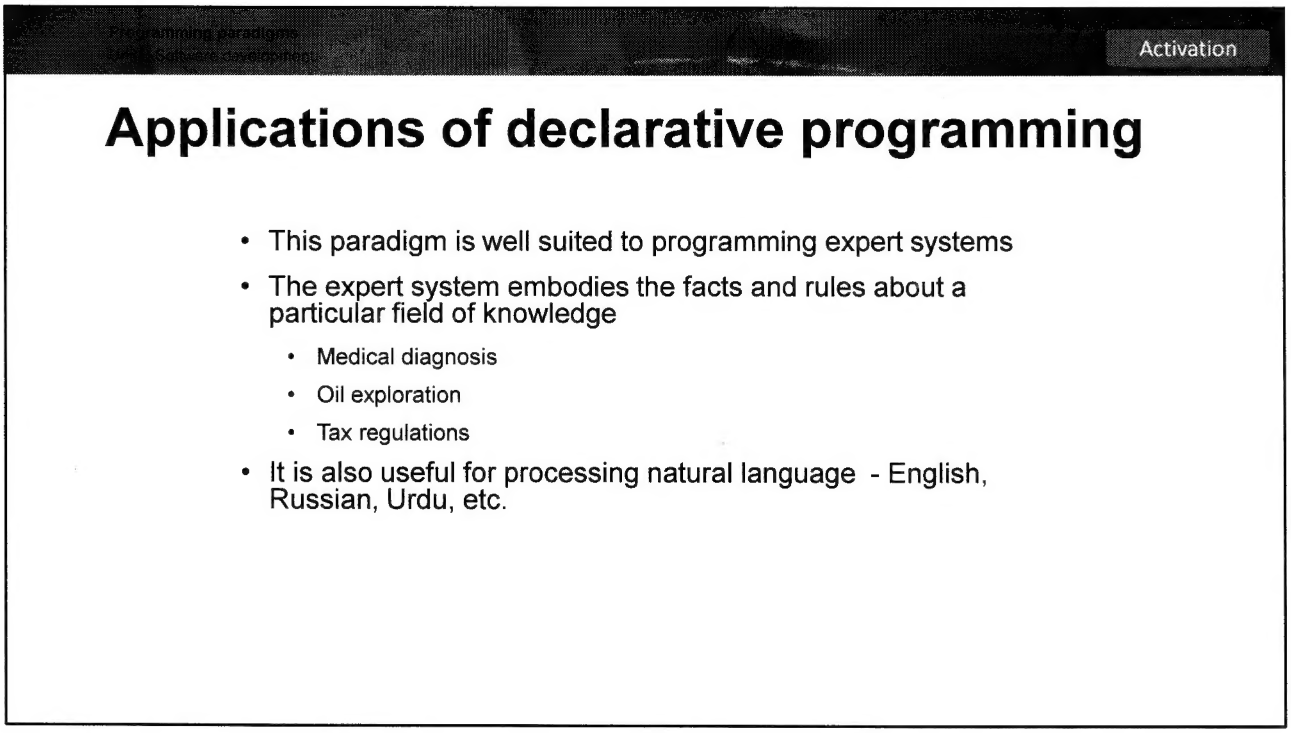
- Instead of defining how a problem is to be solved, the programmer states the facts and rules associated with the problem
- A fact is always unconditionally true, and a rule is true depending on a given condition
  - The order in which the facts and rules are stated is not important, so it is easy to add, change or delete rules
- Executing a program involves stating a goal to be achieved and allowing Prolog to determine whether that goal can be achieved using given facts and rules



Programming paradigms
Software development
Activation
PG ONLINE

Backtracking

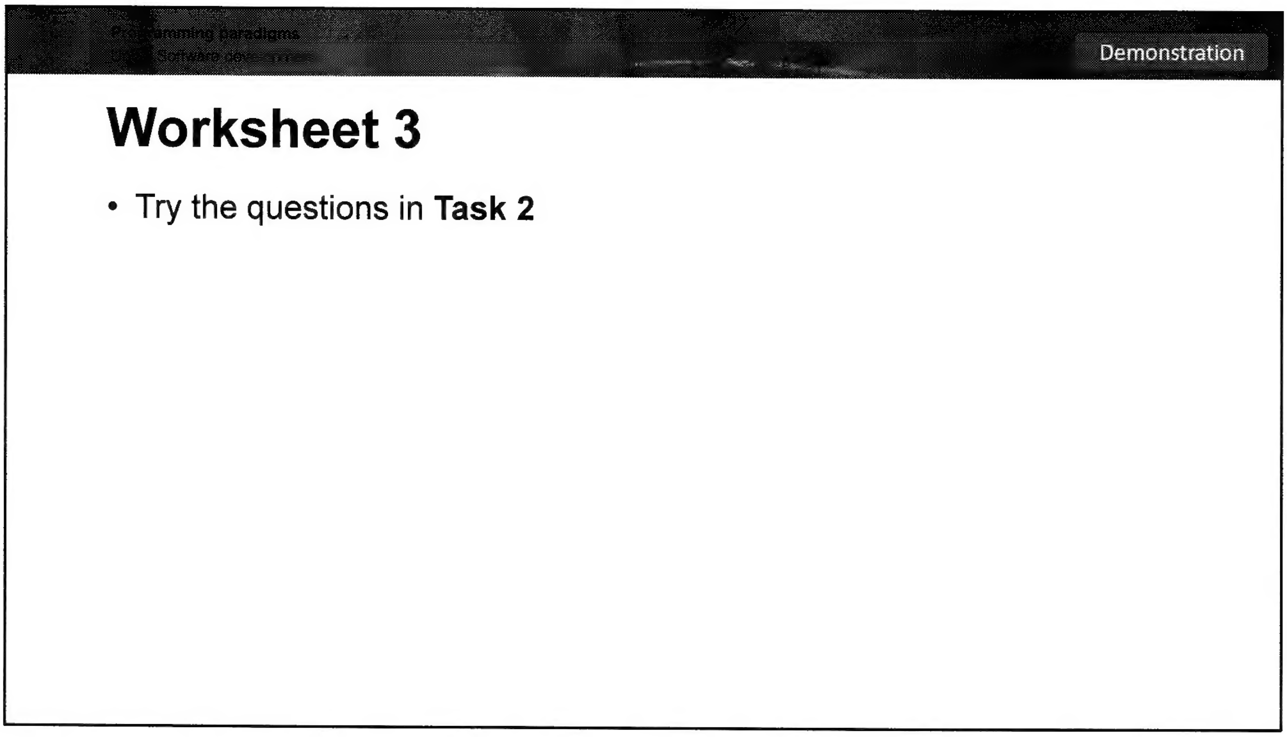
- Given a problem to solve, Prolog selects a route through the maze of facts and rules
- Like Theseus and his ball of string in the Minotaur's maze, it can always find its way back if that proves to be a dead end
  - It will **backtrack** to the last decision point and try another route until either the goal is achieved or there are no further routes to try
  - **Backtracking** is an important feature of declarative programming



Programming paradigms
Software development
Activation

Applications of declarative programming

- This paradigm is well suited to programming expert systems
- The expert system embodies the facts and rules about a particular field of knowledge
  - Medical diagnosis
  - Oil exploration
  - Tax regulations
- It is also useful for processing natural language - English, Russian, Urdu, etc.

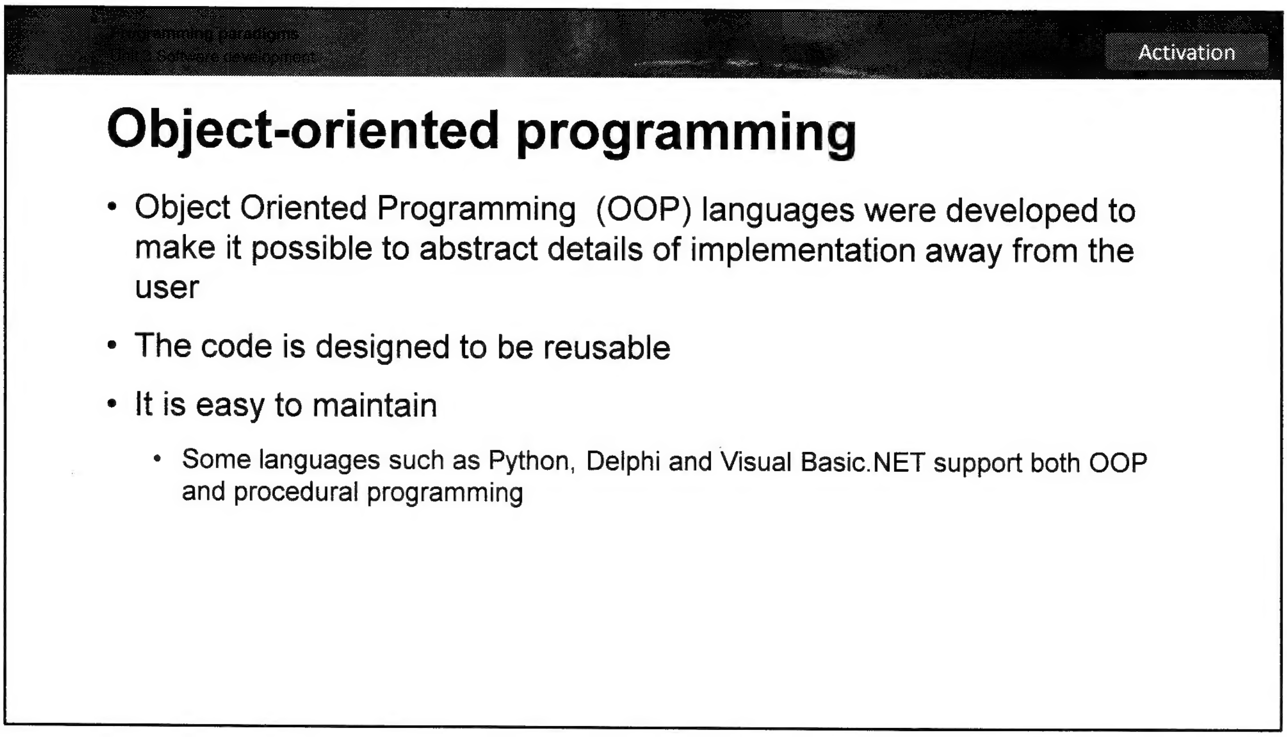


Programming paradigms
Unit 2 Software development

Demonstration

Worksheet 3

- Try the questions in **Task 2**

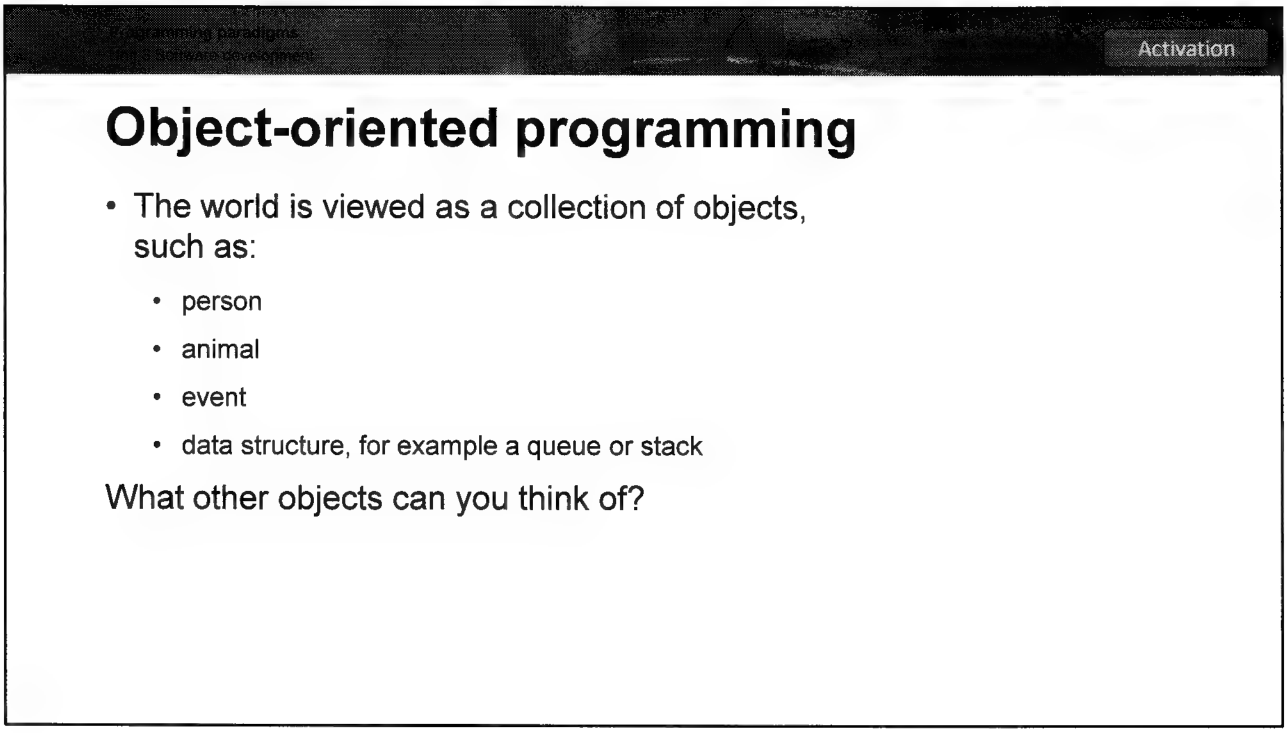


Programming paradigms
Unit 2 Software development

Activation

Object-oriented programming

- Object Oriented Programming (OOP) languages were developed to make it possible to abstract details of implementation away from the user
- The code is designed to be reusable
- It is easy to maintain
  - Some languages such as Python, Delphi and Visual Basic.NET support both OOP and procedural programming



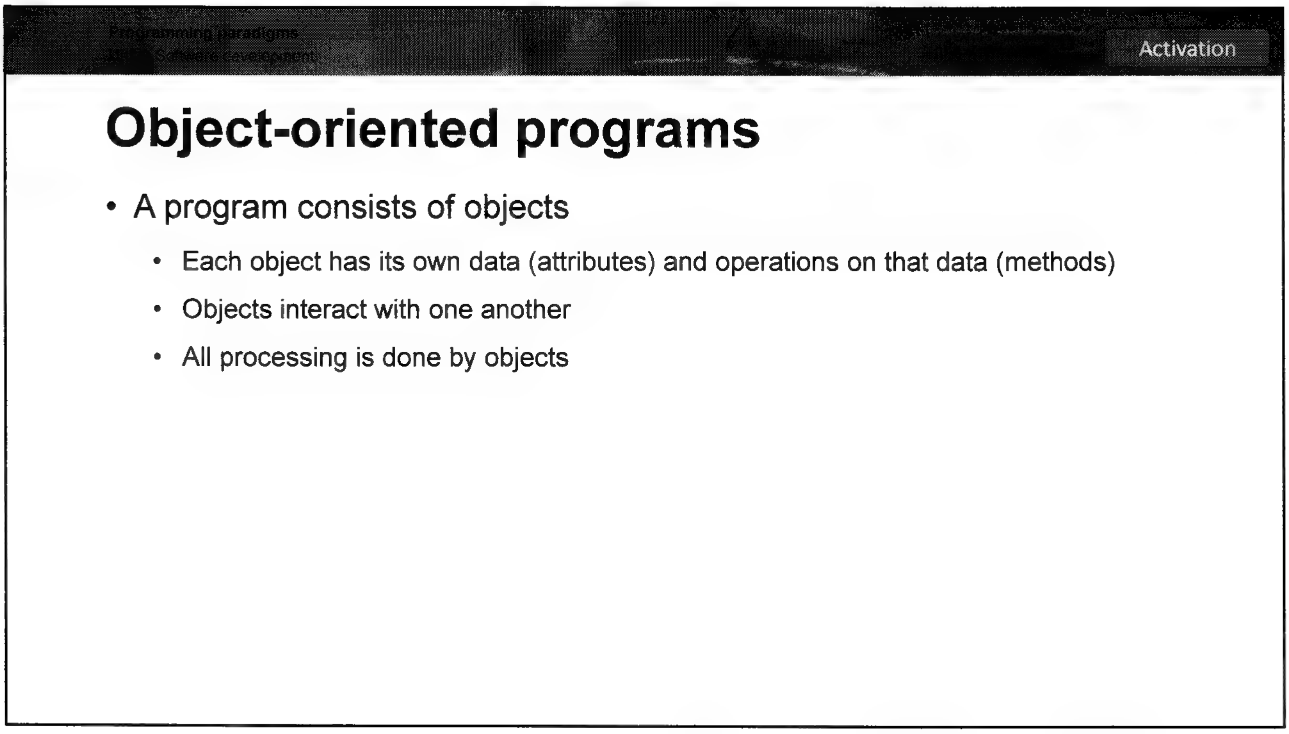
Programming paradigms
Unit 5 Software development

Activation

Object-oriented programming

- The world is viewed as a collection of objects, such as:
  - person
  - animal
  - event
  - data structure, for example a queue or stack

What other objects can you think of?

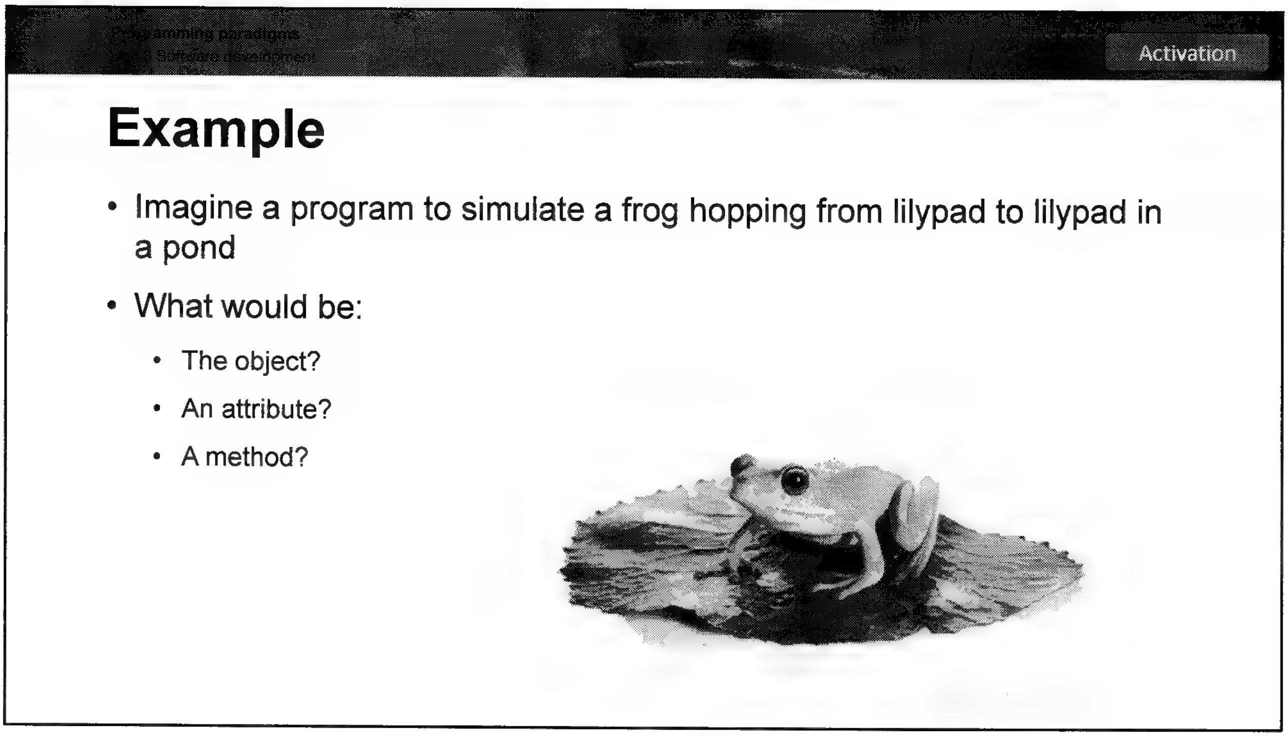


Programming paradigms
Unit 5 Software development

Activation

Object-oriented programs

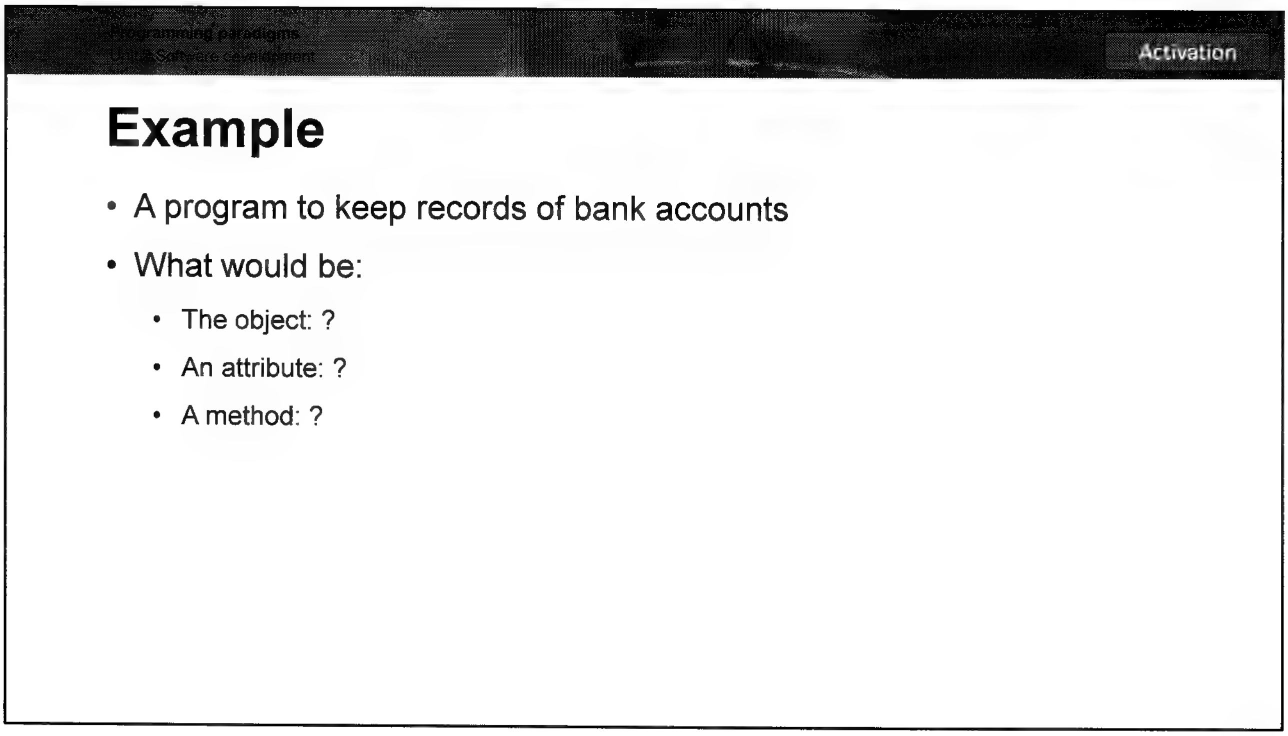
- A program consists of objects
  - Each object has its own data (attributes) and operations on that data (methods)
  - Objects interact with one another
  - All processing is done by objects



Activation

Example

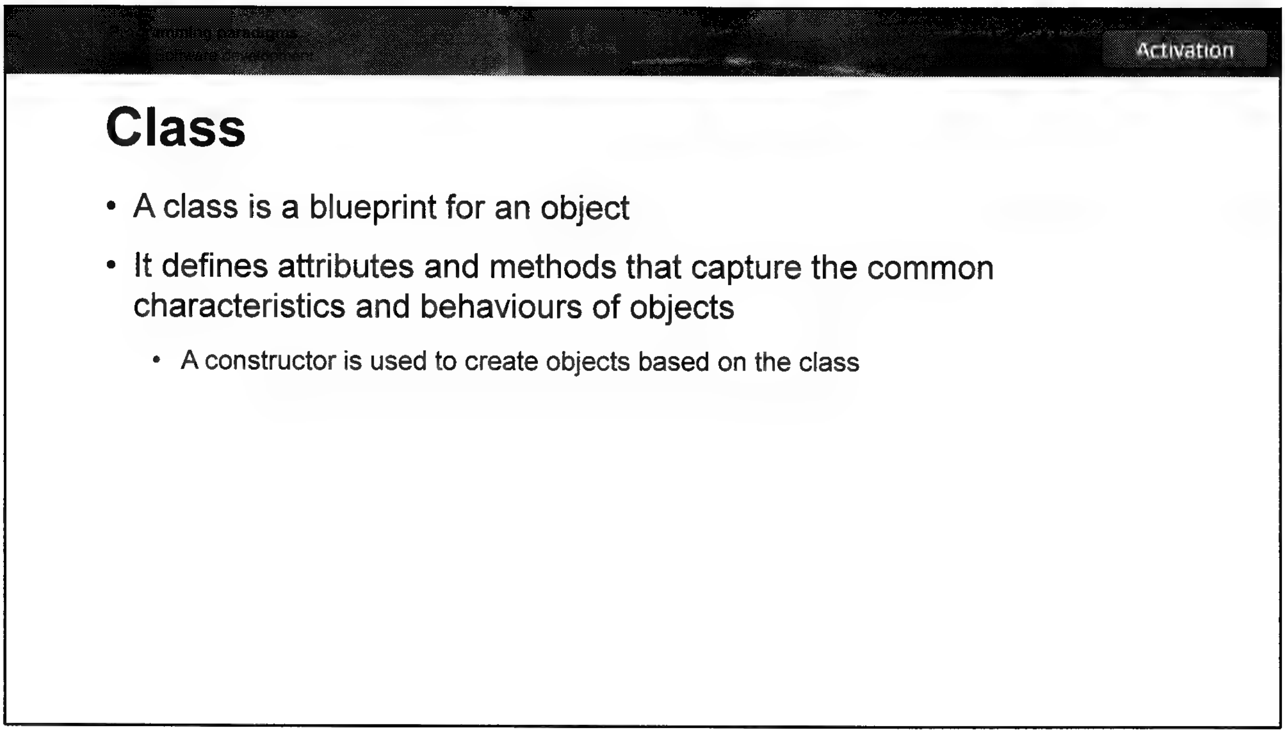
- Imagine a program to simulate a frog hopping from lilypad to lilypad in a pond
- What would be:
  - The object?
  - An attribute?
  - A method?



Activation

Example

- A program to keep records of bank accounts
- What would be:
  - The object: ?
  - An attribute: ?
  - A method: ?

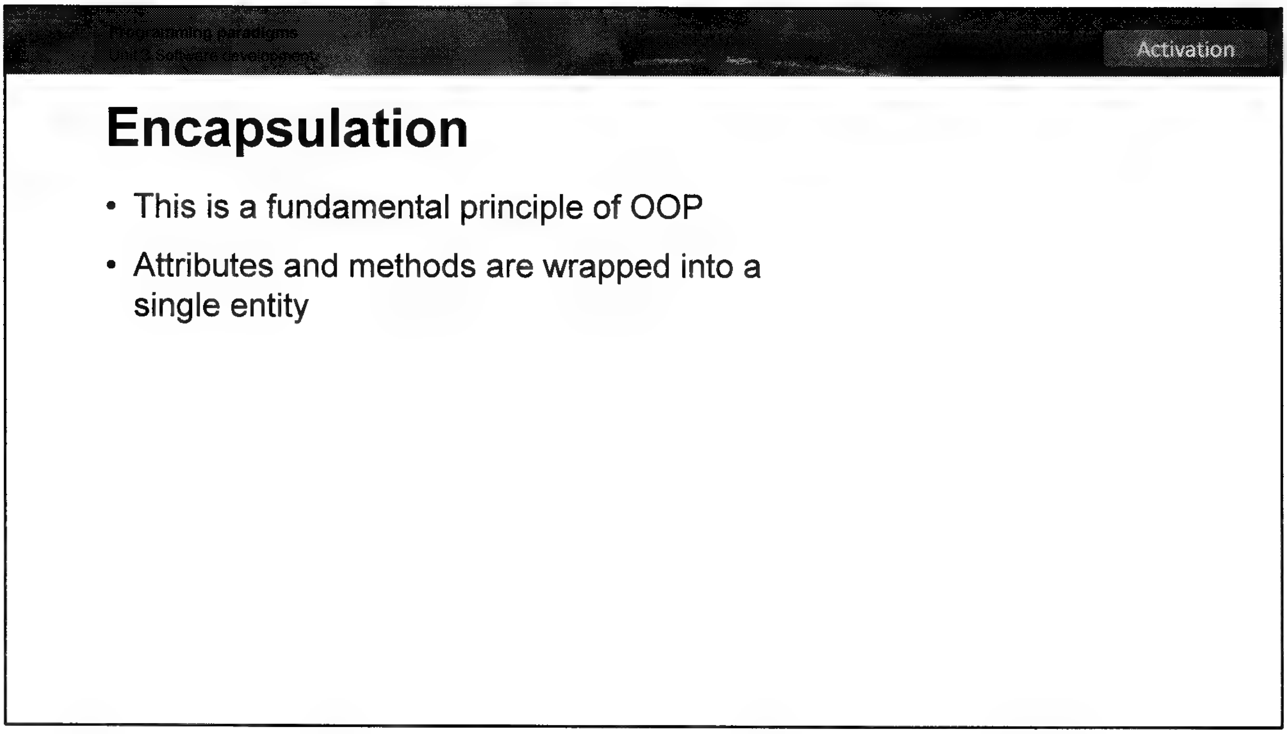


Programming paradigms
Unit 3 Software development

Activation

Class

- A class is a blueprint for an object
- It defines attributes and methods that capture the common characteristics and behaviours of objects
    - A constructor is used to create objects based on the class

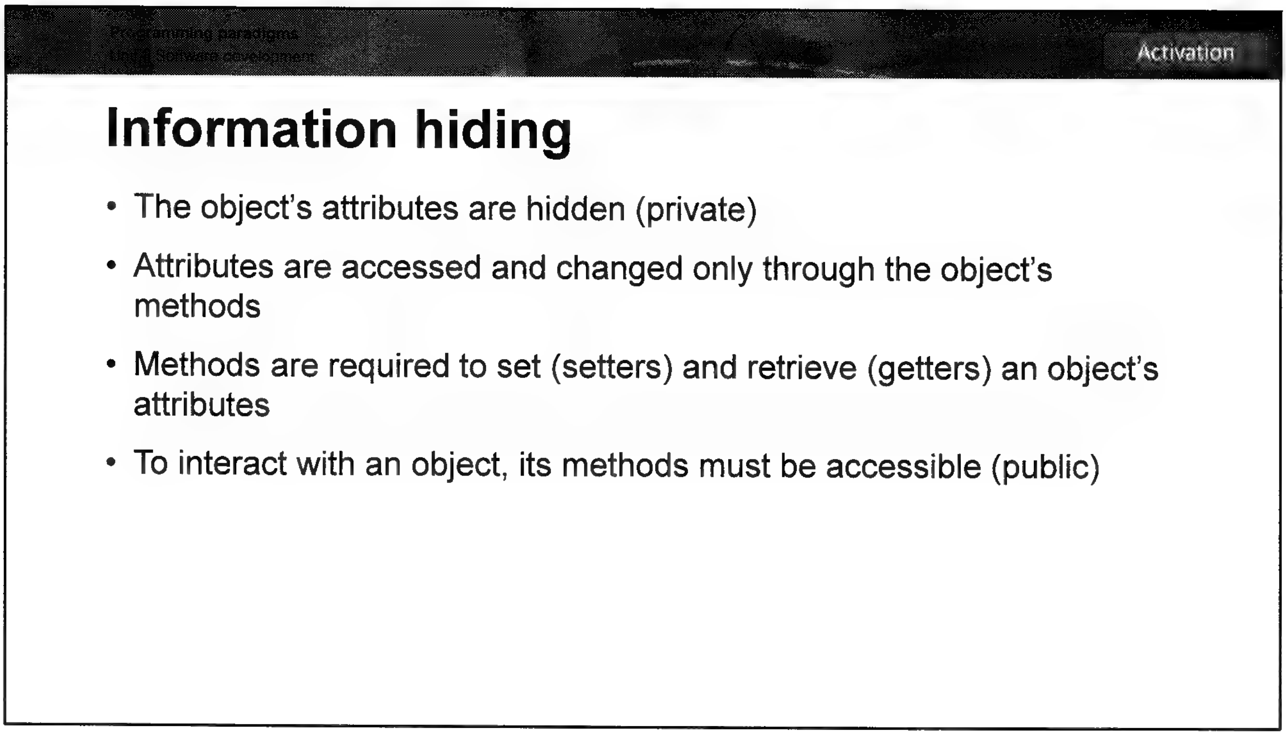


Programming paradigms
Unit 3 Software development

Activation

Encapsulation

- This is a fundamental principle of OOP
- Attributes and methods are wrapped into a single entity

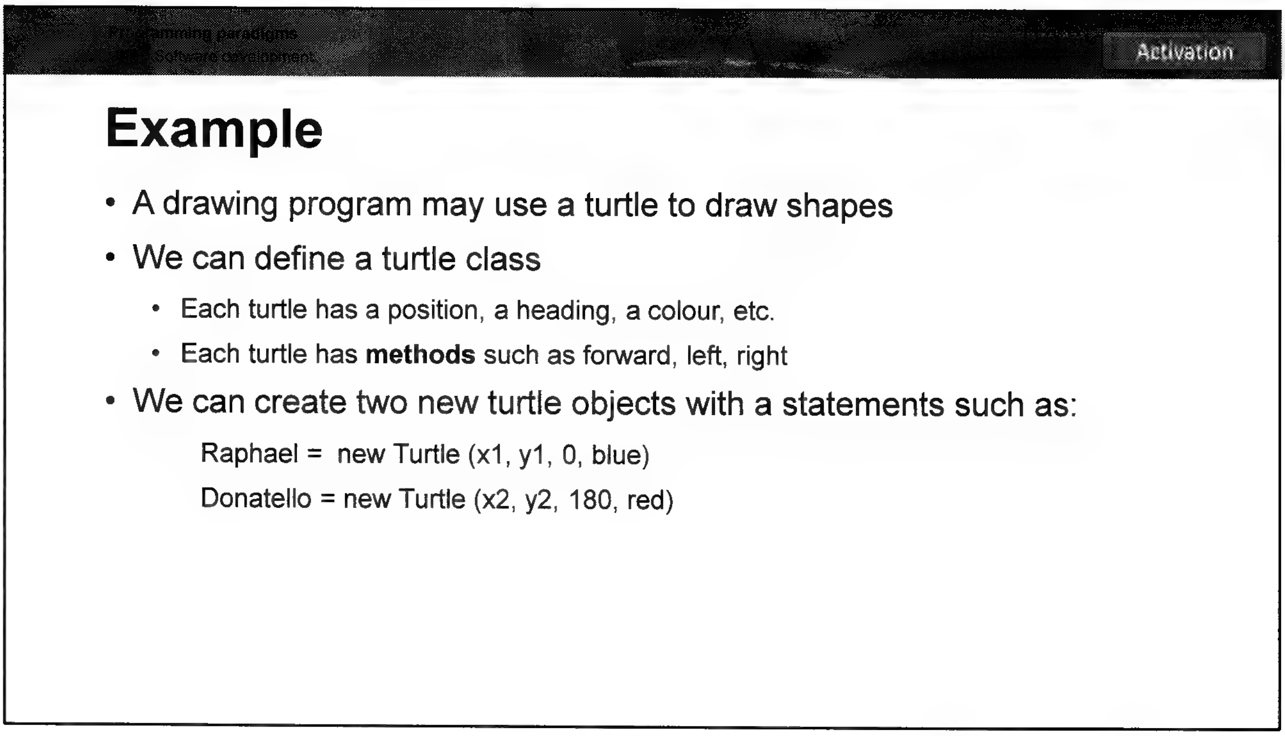


Programming paradigms
Unit [illegible] Software development

Activation

Information hiding

- The object's attributes are hidden (private)
- Attributes are accessed and changed only through the object's methods
- Methods are required to set (setters) and retrieve (getters) an object's attributes
- To interact with an object, its methods must be accessible (public)



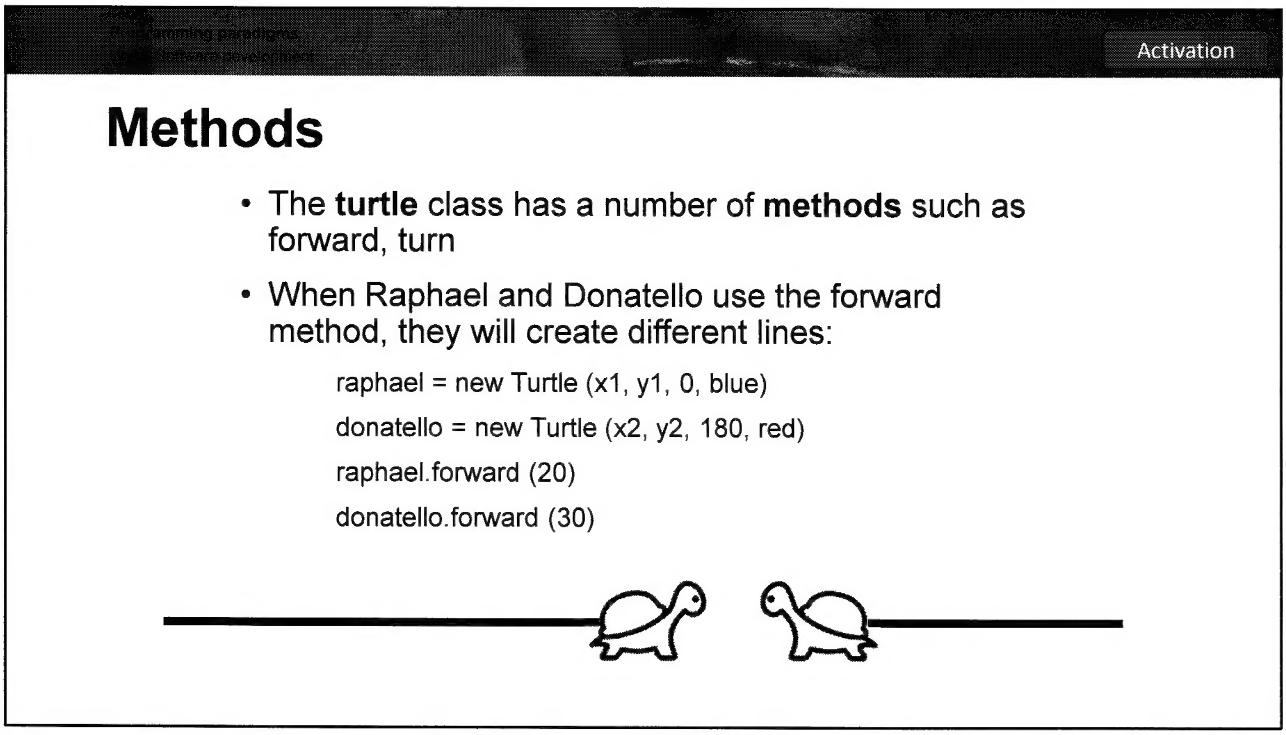
Programming paradigms
Unit [illegible] Software development

Activation

Example

- A drawing program may use a turtle to draw shapes
- We can define a turtle class
  - Each turtle has a position, a heading, a colour, etc.
  - Each turtle has **methods** such as forward, left, right
- We can create two new turtle objects with a statements such as:

```
Raphael =  new Turtle (x1, y1, 0, blue)
Donatello = new Turtle (x2, y2, 180, red)
```



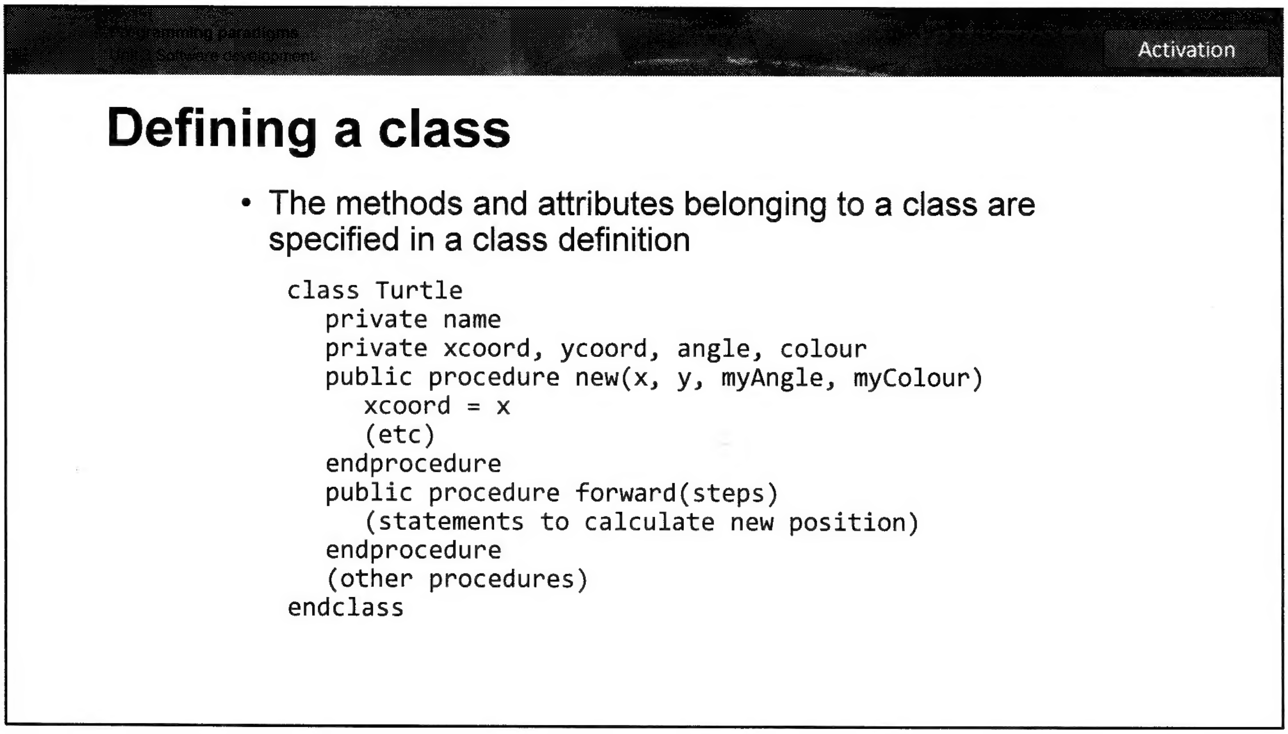
Programming paradigms
Unit 1 Software development

Activation

Methods

- The **turtle** class has a number of **methods** such as forward, turn
- When Raphael and Donatello use the forward method, they will create different lines:

```
raphael = new Turtle (x1, y1, 0, blue)
donatello = new Turtle (x2, y2, 180, red)
raphael.forward (20)
donatello.forward (30)
```



Programming paradigms
Unit 1 Software development

Activation

Defining a class

- The methods and attributes belonging to a class are specified in a class definition

```
class Turtle
   private name
   private xcoord, ycoord, angle, colour
   public procedure new(x, y, myAngle, myColour)
      xcoord = x
      (etc)
   endprocedure
   public procedure forward(steps)
      (statements to calculate new position)
   endprocedure
   (other procedures)
endclass
```

Inheritance

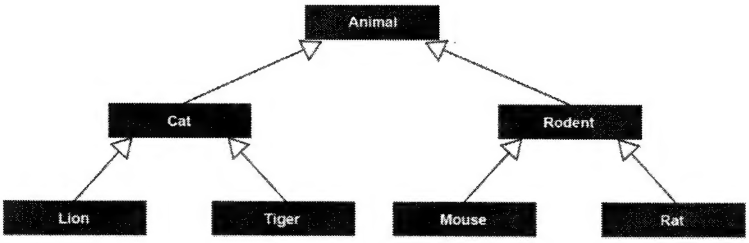
- Objects may be related to other objects in some way
- e.g. a cat and a rodent are both types of animal

Inheritance: a relationship among classes
where a sub-class shares all of the attributes
and methods of a parent class

- Each of the classes cat and rodent will inherit the attributes and methods of the Animal class
- They may, in addition, each have their own attributes and methods

The "is a" rule

- There is a simple rule to determine whether inheritance is appropriate in a program
- Ask: "Is object A an object B?
  - For example, **is a** cat an animal? **Is a** mouse a rodent?



Activation

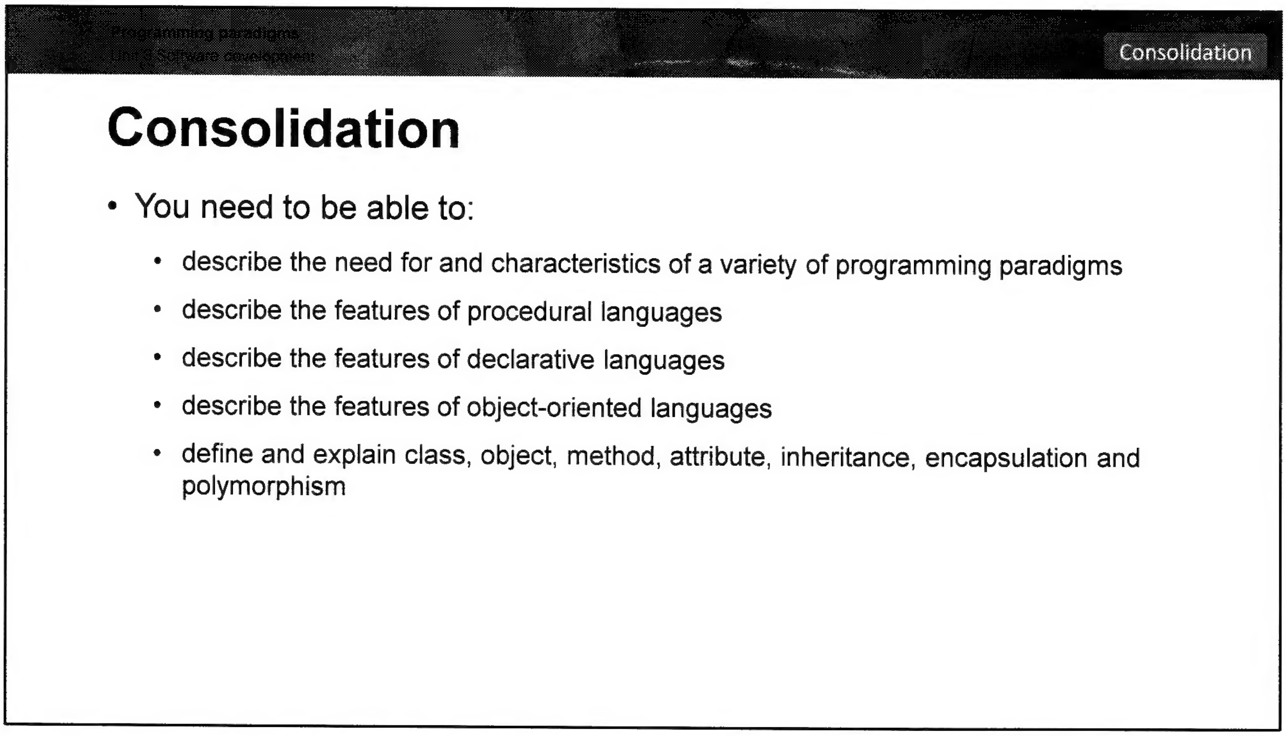
Polymorphism

- An inherited class may have methods and attributes that do not exist in the parent class
- In addition, it may redefine methods that are defined the parent class
- For example, a parent class Bird may have a method **eat**
  - A subclass Parrot may define this as eating seeds
  - A subclass Eagle may define this as eating meat
  - A subclass Chicken may define this as eating both meat and seeds

Demonstration

Worksheet 3

- Try **Task 3** on **Worksheet 3**



Programming paradigms
Unit 3 Software development

Consolidation

Consolidation

- You need to be able to:
  - describe the need for and characteristics of a variety of programming paradigms
  - describe the features of procedural languages
  - describe the features of declarative languages
  - describe the features of object-oriented languages
  - define and explain class, object, method, attribute, inheritance, encapsulation and polymorphism